Bio 264 Homework #2 Solutions

Nicholas J. Gotelli

7 February 2016

> As always, there are many ways to answer these questions in R and in Markdown. Study these solutions for any of the problems that gave you trouble. - NJG

Create a new markdown file called `<Your last name>_HW#2_04Feb2016`. Cut and paste the problems below into Markdown, then add chunks of R code to provide your answers. You will turn in just the `.Rmd` file for this homework assignment.

Some of these problems are adapted from:

> Jones, O., R. Maillardet, and A. Robinson. 2009. Scientific Programming and Simulation Using R. CRC Press, Boca Raton.

1. Suppose $x = 1.1$, $a = 2.2$, and $b = 3.3$. First, reproduce each of the following equations in markdown (by setting them within the `$$` boundaries). Next, assign each expression to the value of the variable z and print the value stored in z.

   a) x^{a^b}
   b) $(x^a)^b$
   c) $3x^3 + 2x^2 + 1$
   d) The digit in the second place of z (hint: use `floor()` and/or `&&`)

Here are the equations as you would type them in Markdown:

```
a) $x^{a^b}$
b) $(x^{a})^b$
c) $3x^3 + 2x^2 + 1$
```

And here is the R code for these problems

```r
x <- 1.1
a <- 2.2
b <- 3.3

z <- x^(a^b)
print(z)
```

```
## [1] 3.61714
```

```r
z <- (x^a)^b
print(z)
```

```
## [1] 1.997611
```

```r
z <- 3*x^3 + 2*x^2 + 1
print(z)
```

```
## [1] 7.413
```

The last problem (d) is a little tricky. We want the digit in the second place of z. At this point, $z = 7.413$, so we are asking R to pull out the digit in the second place, which is a "4". We will start by truncating the decimal piece, then multiplying by 10 and truncating again. That should leave us with the second digit:

```r
# Divide z by its integer component and keep the remainder
z1 <- z %% trunc(z)
print(z1)
```

```
## [1] 0.413
```

```r
# Multiply by 10 and then drop the remainder gives answer
z2 <- trunc(z1*10)
print(z2)
```

```
## [1] 4
```

```r
# If we want to be compact and do this in one calculation:

trunc((z %% trunc(z))*10)
```

```
## [1] 4
```

2. Using the `rep` and `seq` functions, create the following vectors:

   a) $(1, 2, 3, 4, 5, 6, 7, 8, 7, 6, 5, 4, 3, 2, 1)$
   b) $(1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5)$
   c) $(5, 4, 4, 3, 3, 3, 2, 2, 2, 2, 1, 1, 1, 1, 1)$

```r
c(1:8,7:1)
```

```
##  [1] 1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
```

```r
rep(1:5,1:5)
```

```
##  [1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
```

```r
rep(5:1,1:5)
```

```
##  [1] 5 4 4 3 3 3 2 2 2 2 1 1 1 1 1
```

3. Create a vector of two random uniform numbers. In a spatial map, these can be interpreted as x and y coordinates that give the location of an individual (such as a marked forest tree in a plot that has been mapped). Using one of R's inverse trigonometry functions (`asin()`, `acos()`, or `atan()`), convert these numbers into polar coordinates (If you don't know what polar coordinates are, read about them on the web or in your calculus textbook).

```r
# Create and print the Cartesian coordinates
xyCoors <- runif(2)
print(xyCoors)
```

```
## [1] 0.1065243 0.8642382
```

```r
# Create a vector to hold the polar coordinates
polarCoors <- vector(mode="numeric",length=2)

# Calculate the vector length
polarCoors[1] <- sqrt(xyCoors[1]^2 + xyCoors[2]^2)

# Calculate the vector angle
polarCoors[2] <- atan(xyCoors[2]/xyCoors[1])

# Print the Polar coordinates
print(polarCoors)
```

```
## [1] 0.8707784 1.4481568
```

4. Suppose that `queue <- c("sheep", "fox", "owl", "ant")` and that `queue` represents the animals that are lined up to enter Noah's Ark, with the sheep at the front of the line. Using R expressions, update the queue successively as

   a) the serpent arrives;
   b) the sheep enters the ark;
   c) the donkey arrives and talks his way to the front of the line;
   d) the serpent gets impatient and leaves;
   e) the owl gets bored and leaves;
   f) the aphid arrives and the ant invites him to cut in line.
   g) Finally, determine the position of the aphid in the line.

```r
queue <- c("sheep", "fox", "owl", "ant")

# a) the serpent arrives;
queue <- c(queue,"serpent")
print(queue)
```

```
## [1] "sheep"   "fox"     "owl"     "ant"     "serpent"
```

```r
# b) the sheep enters the ark;
queue <-queue[-1]
print(queue)
```

```
## [1] "fox"     "owl"     "ant"     "serpent"
```

```r
# c) the donkey arrives and talks his way to the front of the line;

queue <- c("donkey",queue)
print(queue)
```

```
## [1] "donkey"  "fox"     "owl"     "ant"     "serpent"
```

```
# d) the serpent gets impatient and leaves;
queue <- queue[-length(queue)]
print(queue)
```

```
## [1] "donkey" "fox"    "owl"    "ant"
```

```
# e) the owl gets bored and leaves;
queue <- queue[queue !="owl"]
print(queue)
```

```
## [1] "donkey" "fox"    "ant"
```

```
# f) the aphid arrives and the ant invites him to cut in line.
queue <-c(queue[-length(queue)],c("aphid","ant"))
print(queue)
```

```
## [1] "donkey" "fox"    "aphid"  "ant"
```

```
# g) Finally, determine the position of the aphid in the line.
which(queue=="aphid")
```

```
## [1] 3
```

5. Use R to create a vector of all of the integers from 1 to 100 that are not divisible by 2, 3, or 7.

```
z <- seq(1,100)

# subset by using the remainder function %% and the union %
z <-z[z %% 2 != 0 & z%%3 !=0 & z%%7 !=0]
print(z)
```

```
##  [1]  1  5 11 13 17 19 23 25 29 31 37 41 43 47 53 55 59 61 65 67 71 73 79
## [24] 83 85 89 95 97
```

6. Create a vector `z` of 1000 random uniform numbers.
    a) create a vector that contains 3 numbers: the proportion of the numbers in z that are less than 0.10, greater than 0.90, and between 0.45 and 0.55.

```
z <- runif(1000)
prop <- c(mean(z<0.10),mean(z>0.90),mean(z>0.45 & z<0.55))
print(prop)
```

```
## [1] 0.095 0.094 0.094
```

b) Making successive copies of z, transform your vector of uniform numbers in the following ways:

- log (base 10) of z
- z^2
- e^z
- square root of z

c) for each case calculate your vector of 3 numbers to get the new proportions.

d) typeset the formulas from (b) in markdown (with the $$ brackets).

```
#log10 of z
z <- log10(z)
prop <- c(mean(z<0.10),mean(z>0.90),mean(z>0.45 & z<0.55))
print(prop)
```

```
## [1] 1 0 0
```

```
# z^2
z <- z^2
prop <- c(mean(z<0.10),mean(z>0.90),mean(z>0.45 & z<0.55))
print(prop)
```

```
## [1] 0.526 0.108 0.029
```

```
# e^z
z <- exp(z)
prop <- c(mean(z<0.10),mean(z>0.90),mean(z>0.45 & z<0.55))
print(prop)
```

```
## [1] 0 1 0
```

```
# sqrt(z)
z <- sqrt(z)
prop <- c(mean(z<0.10),mean(z>0.90),mean(z>0.45 & z<0.55))
print(prop)
```

```
## [1] 0 1 0
```

To type set this in Markdown, use:

```
$$ log_{10}(z) $$

$$ z^2 $$

$$ e^z $$

$$ \sqrt{z} $$
```

$$log_{10}(z)$$

$$z^2$$

$$e^z$$

$$\sqrt{z}$$